Automated Sketching and Engineering Culture

Peter Varley and Pedro Company
Dept. of Mechanical Engineering and Construction. Universitat Jaume I (Spain)
<varley@emc.uji.es>,<pcompany@emc.uji.es>

Abstract

In this paper we argue that sketching is an essential part of engineering culture. If current CAD tools cannot support sketching, it is the tools, not the culture, which must change.

We then consider how close we are to having Computer-Aided Sketching tools which meet the requirements of traditional engineering culture. This discussion considers three categories of sketch: thinking sketches, used to focus and guide non-verbal thinking; talking sketches, employed to support discussion of the design with colleagues; and prescriptive sketches, which give instructions to the draftsman in charge of making the finished drawing.

1. Introduction

It has been said that "*The real danger is not that computers will begin to think like men, but that men will begin to think like computers*" (the quotation is attributed to S.J. Harris). It has also been said that "*If you have a hammer, every problem looks like a nail*" (the quotation is anonymous). How we work, and even how we think, is constrained by the tools we habitually use.

In the past, engineering designers made much use of pencil and paper, those being the tools they had available. As a result, the ability to create and understand sketches became a necessary part of engineering culture. We discuss the advantages and limitations of pencil and paper sketching in Section 2.

It is clear that use of current CAD tools can overcome some of the limitations of pencil and paper sketching. It is, unfortunately, also the case that use of current CAD tools also loses many of the advantages of pencil and paper sketching. We discuss this too in Section 2.

If engineering culture were simply to accept that the ability to use CAD tools is an *additional* necessary skill, there would be no problem. However, there are those (e.g. [3]) who argue that CAD can *replace* sketching, that sketching should be considered old-fashioned and that it can be discarded from the set of necessary engineering skills. We disagree entirely with this latter viewpoint.

Ideally, what we should want would be tools which combine the advantages of sketching with those of CAD. The rest of this paper discusses how close we are to creating such tools. Where specific examples are required, we focus on work to which we have made personal contributions, while noting those various places where the work of others is in advance of our own.

Unlike Olsen et al [27], which classifies sketch-based tools as viewed by the tool-maker (e.g. by data types and operations supported), we prefer to subdivide our survey using a classification proposed by Ferguson [8] which considers them from the point of view of the user: Section 3 discusses *thinking sketches*, used to focus and guide non-verbal thinking; Section 4 discusses *talking sketches*, employed to support discussion of the design with colleagues; and Section 5 discusses *prescriptive sketches*, which give instructions to the draftsman in charge of making the finished drawing.

Finally, Section 6 summarises our conclusions and highlights the most pressing areas for future work.

2. Sketching: Pros and Cons

"*The pencil has been called the most potent instrument in the world, for it gives most of man's thought and aspirations their first visible form*" [2].

Ferguson [8] notes that two designers never just sit down and talk. They draw sketches for one another. Designers even take the pencil from one another as they talk and draw together on the same sketch. Such sketches will continue to be important in the process of going from vision to artefact, as they make it easier to explain a technical point, because all parties in a discussion share a common representation of the idea being debated.

Recent studies, e.g. Chamorro et al [4], have show that visual representations of concepts are also useful for enhancing communication between users and designers, by enabling identification of differences between designers' and users' concepts of the product.

The Engineering Design Graphics Division of the American Society for Engineering Education (ASEE) identifies the ability to sketch engineering objects freehand as the second most important skill to be learnt by students of engineering [1]. This conclusion is supported by the American Society of Mechanical Engineers (ASME) [31].

Teachers of engineering design continue to produce textbooks which instruct their students in the techniques of creating and understanding sketches (Lieu and Sorby [18] is a recent addition to this literature).

We can conclude from this that sketching is a vital part of the culture and tradition of engineering design.

However, graphical literacy has recently suffered a decline. New engineers often have inadequate experience in making sketches by hand in order to effectively communicate information graphically. As a result of using computer-aided drafting in both engineering education and professional practice, hand-sketching skills have been overlooked ([12,22,31]). Also, some engineers when faced with conceptual design now rely more on verbal and numerical synthesis tools than on graphical ones [38] since the scope of graphical tools is supposed to be limited to detailed design and manufacturing specification.

The main weakness of paper-and-pencil sketching comes from the fact that after a final sketch has been obtained and the conceptual design is complete, the designer must create the CAD model from scratch. Important as it is, sketching activity cannot remain disconnected from the rest of the design process.

However, commercially available CAD tools with some "pseudo-sketching" capabilities are far from being a satisfactory alternative to sketching. Current CAD tools push designers in precisely the wrong direction. As Jonson [12] says: "*the computer encourages the user to go straight into the finished work without the critical and creative thought period*".

Such a process is far from being genuine sketching, where incomplete and ill-defined geometries actually germinate from the designer's mind's eye.

It is, perhaps, for this reason that many designers still use paper-and-pencil sketches, and any tools we offer them must offer them significant additional functionality in addition to duplicating all of the functionality which they already enjoy. We identify, as significant advantages of CAS over paper-and-pencil: easy storage and transfer; limitless drawing space (zooming and other virtual-paper navigation tools convert the limited screen surface into limitless virtual paper); and edit capabilities (erasing, copying, resizing and other transforming operations help to convert the virtual-pen into a more effective tool than the conventional pen). Finally, digital representation allows the sketches to be integrated into a product lifecycle management system.

3. Thinking Sketches

Conceptual design is a complex process and one which is poorly understood [29]. In a sense, design is a process of discovery [11]: it is the refinement of thought by means of visual imagery.

Sketches are more useful than line-drawings during this thinking process, since a designer can focus on the creative aspects of his work, not spend time on routine aspects such as managing rule and compass, drawing auxiliary construction or calculating coordinates and entering data into a CAD package. The visual feedback of a drawing enables an experienced designer to see instantly what his sketch implies, and if he does not like it he can draw something else without having wasted any time.

Menezes et al. [25] observe that interaction with drawings seems to be even more relevant to designers than the physical skill of drawing. As a result, the simple "what you draw is what you imagine" interface provided by pencil and paper continues to be useful as it is less distracting than a set of drawing devices, and will continue to be useful if it is provided by a pen and tablet (rather than by an array of menus and icons).

In this section we only consider recent developments. For older work, we refer the reader to surveys by Wang and Grinstein [39] and Company et al [6]. Of the variety of approaches which these surveys discuss, essentially only two satisfy our requirement for a natural sketching interface: interpretation of wireframe drawings (drawings where hidden lines are shown), and, most natural of all, interpretation of natural line drawings (drawings which show only those lines visible from a particular viewpoint).

3.1 Design Intent

One problem which must be addressed regardless of the type of drawing processed is the problem of design intent. What object did the sketcher have in mind when creating the sketch?

For example, Figure 1 shows a simple example of the design intent problem. Clearly, if the user intended to draw a cube, the central vertex is misplaced. Was this, or was this not, deliberate?

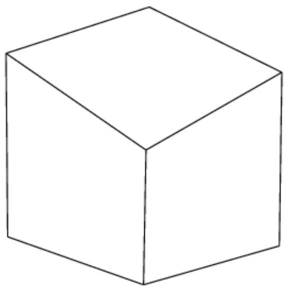

Figure 1: Misplaced Vertex?

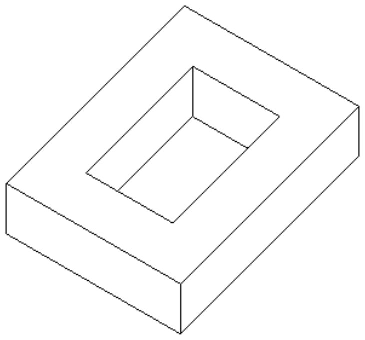

Figure 2: Tray or Ring?

Figure 2 shows a more subtle example, and one which could realistically occur in practice. The height of the central feature is slightly less than the height of the bounding box. Is this, or is this not, deliberate? Depending on which interpretation we choose, we get a different object. If we assume that the difference was deliberate, the central feature is a pocket, and the object is a tray. If we assume that the difference was accidental, the central feature is a through hole, and the object is a ring.

Although these two figures show natural line drawings, it can readily be seen that the same problems would exist when trying to interpret the corresponding wireframe drawings.

So how is it possible, just by analysis of pencil strokes, to determine what was in a designer's mind?

In practice, by making assumptions about engineering objects and the ways people see and depict them, it is often possible to reproduce a single object which humans will agree is the best interpretation of the drawing.

We believe that we should assume certain regularities whenever it is reasonable to do so. These regularities should be those which are readily perceived, chiefly perpendicularity and symmetry. In addition to the cognitive paradigm that *if the user sees them, they should be there*, some regularities may have a particular role in applications such as structural analysis. Symmetry is the most obvious case. The importance of symmetry in engineering design is well-known (see, for example, [33]), and one classic use of symmetry is to reduce the computational effort in analysis of structures by simplifying the models.

3.2 Wireframe Drawings

We are now close to having robust methods for interpretation of wireframe drawings. This is usually done in two stages: identify those loops of edges in the drawing which correspond to faces; and inflate the drawing into 3D while keeping the face loops planar. For the former, the approach of Liu, Lee and Cham [21] is robust and usually adequately fast in practice, the approach of Varley and Company [36] is also robust and faster, and an idea proposed by Li [17] promises to be faster still. For the latter, Martin et al [23] provide a list of useful techniques, many of which were originally suggested by Lipson and Shpitalni [20].

The most serious outstanding problem is the one touched on in the previous section, that of design intent. In particular, which loops of edges should be not merely parallel but coplanar? Piquer et al [28] address this for symmetrical objects, but no robust general solution exists.

Current research now concentrates on interactive interpretation of sketches: what assistance can be given to the sketcher while sketching is in progress?

CIGRO [7] was developed in order to study the feasibility of using progressive interactive interpretation of freehand sketches to create 3D geometry. *CIGRO* interactively reconstructs polyhedral objects from freehand sketches to yield three-dimensional models. In the current implementation, the geometric domain is restricted to quasi-normalons.

One key characteristic of *CIGRO* is its minimalist interface based on only three gestures: create geometry line, create auxiliary line, and delete.

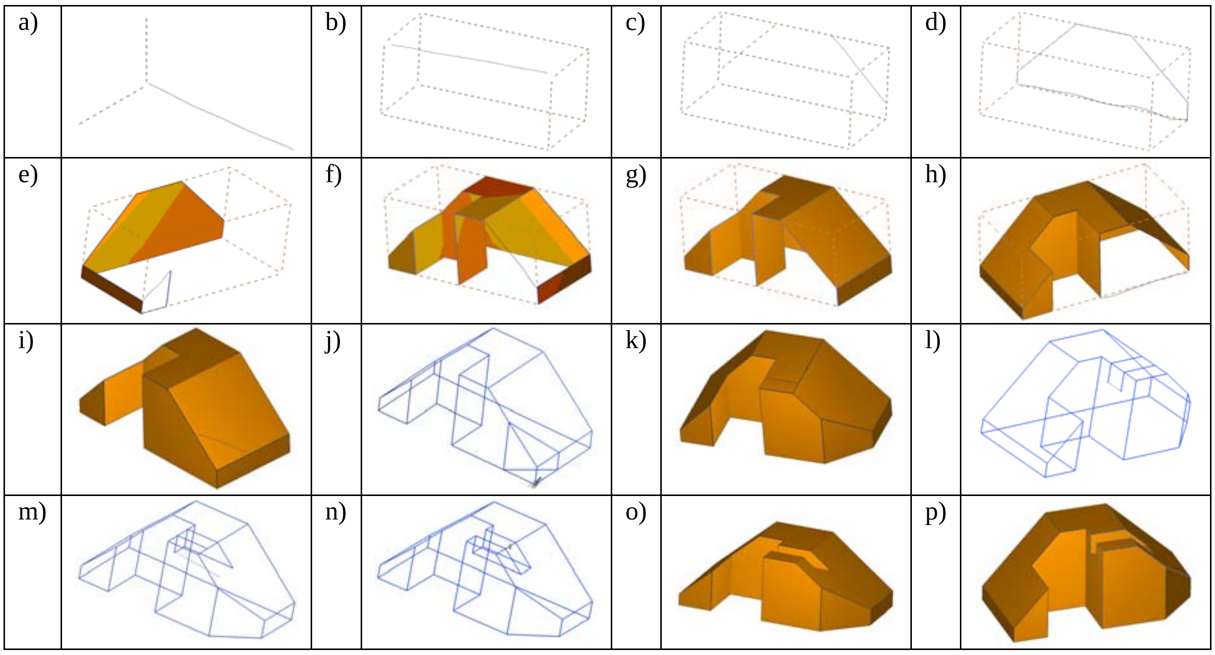

Figure 3. Modelling sequence in CIGRO

The user creates a new object by performing a pseudo-axonometric drawing of it. Usually the first step is the definition of the main directions (Figure 3.a).

CIGRO supports creation of complex objects by blocking them in within a frame of construction lines (in Figure 3.a and 3.b the beautified auxiliary lines are shown as dashed lines). Construction lines, drawn by applying low pressure with the stylus, can then be used for snapping or over-sketching (Figure 3.d) the desired geometry lines. This is common practice in engineering drawing, as blocking in the main features of a part not only simplifies sketch construction, but also allows the creation of complex parts from simple geometry.

CIGRO comprises a simple gesture analyser, a line-drawing beautifier and an axonometric inflation engine.

In order to support interactive operation, the axonometric inflation engine provides tentative 3D models. This allows users to see the current 3D model as they complete the sketch. As soon as the system detects a complete face, it is shaded automatically (see shaded faces from Figure 3.e to 3.h). Users can change the point of view whenever they like, and proceed with sketching (example in Figure 3.d to 3.m). Removal of undesired edges is performed drawing a scratch stroke, which is interpreted as a delete command (as seen in Figures 3.j and 3.n).

We have found in practice that, when they are used to create geometry from scratch, users ask for some kind of assistance with repetitive tasks, such as drawing all the edges in extrusion-like objects, or a symmetry operator to construct symmetric shapes by only drawing half. In order to improve the usability our CAS tools, we should provide such extended drawing commands. In order to maintain the simplicity of CIGRO's user interface, these commands will be implemented as gestures, in the same way that the delete command is currently implemented as a gesture.

Another practical observation (based on both our own experience and that of [19]) is that it is important to remind the user that sketching is tentative, not definitive. To this end, we are currently implementing a non-photorealistic renderer to preserve the "sketchy" look of objects created from freehand sketches. Figure 4 shows an example of an object rendered "sketchily".

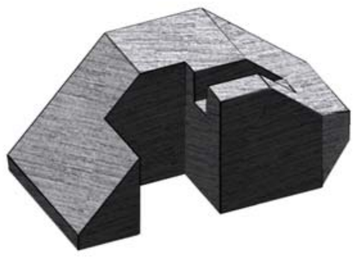

Figure 4. A "Sketchy" Object

3.3 Natural Line Drawing

The problem of interpreting natural line drawings is inherently more difficult than that of interpreting wireframes: what is around the back of the object?

RIBALD [34] is an attempt to interpret natural line drawings as objects, creating a boundary-representation solid model of the object which the drawing portrays. There are more recent programs with more natural user interface than RIBALD (e.g. [16] and [24]), but none of them add anything to the art of interpreting drawings.

The underlying assumption is that the *correct* interpretation of the drawing is the object an engineer would regard as the *most reasonable* interpretation of the drawing. By defining the *correct* interpretation as we do, we place the problem of interpreting drawings firmly in the realm of engineering culture, not of abstract mathematics.

For example, while there are infinitely many objects which *could* result in the drawings in Figure 5, in practice, an engineer would be in little doubt as to which was the correct interpretation.

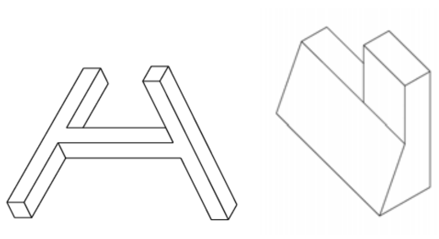

Figure 5. Natural Line Drawings

RIBALD is fully-automatic, requiring no manual intervention except for creation of the line drawing.

The other requirement in producing RIBALD was that the 3D model should be produced quickly: ideally, within a second. Such rapid feedback would not merely remove a bottleneck in the design process but could improve it by allowing designers to refine their ideas while they were still fresh in their minds.

RIBALD interprets line drawings in a three-stage process: (a) create the *frontal geometry*, in which everything visible in the natural line drawing is given a position in 3D space; (b) add the *hidden topology*, the occluded part of the object not visible from the chosen viewpoint, and (c) *beautify* the geometry of the completed object—beautification of solid models is an area of active research in its own right, with applications in other fields such as reverse engineering, and is not discussed further here. RIBALD does not itself convert the engineer's original freehand sketch to a line drawing—this is regarded as a separate problem, considered in Section 3.1 above.

The three most important aspects of frontal geometry are (a) line labels—which lines in the drawing correspond to convex, concave and occluding edges of the frontal geometry; (b) line parallelism-which groups of lines did the designer intend to correspond to parallel edges; and (c) inflation to 2½D by assigning depth coordinates to vertices.

It is found that performing these three tasks sequentially, in any order, presents difficulties, as each yields information useful to the others. For example, given an object which has already been inflated correctly, determining the line labels is trivial, but inflating an object correctly without knowing which vertices touch faces and which vertices occlude faces presents problems. On the other hand, while labelling genus-zero trihedral drawings is trivial (e.g. using Kanatani's algorithm [15]), labelling uninflated drawings with higher-order vertices is unreliable at best [35].

Approaches which iterate these steps, or perform them in parallel, are to be preferred, but even here there remains work to be done before our objectives of speed and robustness are met [37].

RIBALD constructs hidden topology by performing a greedy search through the space of possible additional topology. This is successful in the cases shown in Figure 6. It will be noted that the more restrictive the search domain, the more successful the search—quite complex trihedral normalons can be constructed, but the construction of general (non-trihedral, non-normalon) polyhedra is very limited.

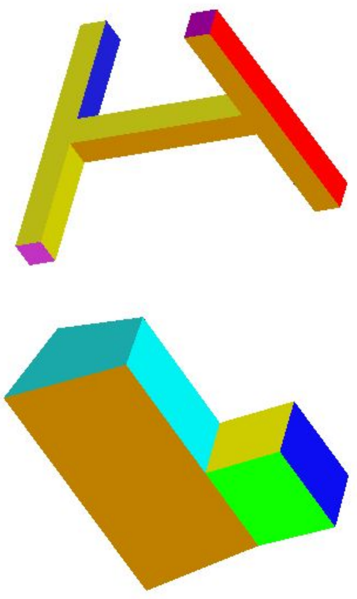

Figure 6: Completed Objects

Recent attempts to improve RIBALD's greedy approach by advance detection and avoidance of error conditions has made little practical difference to the robustness of the method, and it may be necessary to look elsewhere for a good approach.

Grimstead [10] suggested treating planes as half-space separators, faces as patches of planes, and edges as half-space operators (convex edges are "intersection" and concave edges are "union"). Our preliminary investigations found this idea to be hard to implement even for normalons, and the results were unimpressive.An alternative idea, suggested as long ago as Roberts [30] and shown by Suh [32] to be a practical approach to constructing hidden topology, is to use simple polyhedra as half-spaces (Roberts suggested using cuboids; Suh's implementation uses extrusions of polygonal end-caps).

Suh's implementation assumes an accurate drawing. It remains to be seen whether or not it can be extended successfully to freehand drawings, which will inevitably contain drawing errors.It can be noted that Suh's approach is inevitably suboptimal as a design tool. On the one hand, it is not an ideal method of interpreting line drawings, since it is inherently restricted to drawings of objects which can be decomposed into extrusions of polygons. On the other hand, it is not an ideal tool for design engineers, since if we are to model objects as unions and intersections of extrusions we should allow the design engineer to enter the extrusions in 3D the first place rather than insisting that he draw a 2D natural line drawing which the program interprets as extrusions.

Nevertheless, in the absence of any significant improvement to RIBALD's greedy approach, it is Suh's approach which represents the state of the art.

4. Talking Sketches

By definition, the intended audience of *talking sketches* is people, not computers. Why should anyone should be interested in automatic interpretation of talking sketches? We identify two possible answers.

Firstly, as the design process becomes more complex (as do its products!), more and more of the process is recorded. We can safely forecast that, in the near future, talking sketches will be recorded as soon as the required hardware (tablet PCs and the like) becomes commonplace.

Secondly, computers cannot manage data properly without having some ability to interpret it. As soon as recording of talking sketches becomes established practice, interest will grow in how computers can be made to understand them.

Jonson [13] documents some recent developments from the Computer Support Collaborative Work (CSCW) community which are aimed at both collaborative creation and the sharing of 2D sketches.

However, little work has been done in the field of automatic interpretation of talking sketches and converting them into 3D digital models. Although talking sketches are clearly going to be an important area in the near future, there is much work remaining to be done.

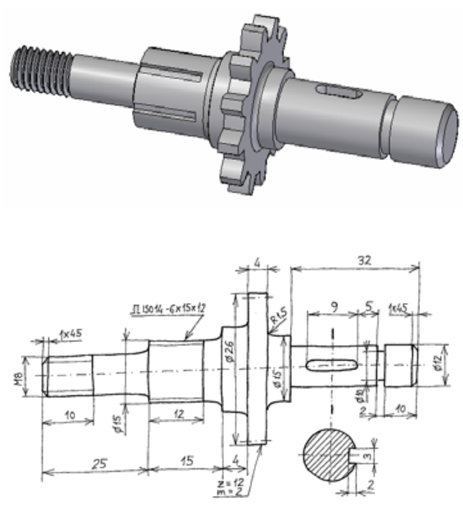

Figure 7. 3D model of a shaft (above) and its prescriptive sketch (below)

5. Prescriptive Sketches

Prescriptive sketches such as the one in Figure 7 are technical drawings which contain detailed information describing a final design. They contain a full set of views complemented by symbolic information conveyed through standardised symbols. They only differ from detailed design drawings in that the views, while containing a detailed description of the shape and a rough estimation of metric properties, need not be geometrically perfect.

Prescriptive sketches include standardised symbolic information (see "technical drawings" at www.iso.ch or www.asme.org) which can be classified into: (a) *Pseudo-textual symbols* which identify very common shapes—the "diameter" symbol (Ø) is the most common; (b) *Graphic symbols* which conventionally simplify and/or highlight a true geometrical feature of the product. Hatching linked to cut views (ISO 128) is the most common of this kind of symbol. Splines and serrations (ISO 6413), gears (ISO 2203), and screw threads and threaded parts (ISO 6410), among others, show the complexity of this class of symbol; and (c) *Multi-iconic symbols* which convey a complex product's features. Dimensions (ISO 129) are by far the most common example of such symbols. Numerous manufacturing symbols (e.g. welding symbols), tolerancing and kinematic diagrams also belong to this category.

In current industrial practice, only pencil and paper prescriptive sketches are used. However, with the spread of CAS environments, prescriptive sketches will play a more important role in design.

Prescriptive sketches are harder to interpret than other types of sketch as they convey richer symbolic information. Automatic interpretation of prescriptive sketches has much in common with automatic interpretation of blueprints, which field is somewhat in advance of our own. There are, however, differences. We note, for example, a recent paper by Gong et al [9] describing a method for segmenting blueprints. This work relies on the geometry of the blueprint data being accurate, and it is not yet clear whether or not it can be transferred to the field of prescriptive sketches.

ParSketch [26] is a prototype application developed in order to investigate and address the problems posed by prescriptive sketches. It implements our minimal interface concept by providing a sketching environment which is free of menus and icons. We are now taking this idea further with a version aimed at sketch-based structural pre-processing [5].

ParSketch interprets strokes which can be recognised as geometry (line, arc, circle, ellipse, or composed entities which are automatically segmented into these basic entities), and symbols representing constraints (dimension, parallel, perpendicular, tangent, concentric, horizontal or vertical). Unwanted drawing entities can be removed using a scratching gesture. This not only allows errors to be corrected, but also enables more complex shapes to be drawn incrementally, by refining simpler forms.

New gestures can be added to the system by providing new samples to the gesture recogniser.

At present the system uses pen pressure to decide whether input corresponds to geometry or to gesture strokes: high pressure means a geometry stroke, while low pressure means that the stroke is interpreted as a gesture. Both geometry and gesture analysers make use of two geometric signatures: the direction and curvature graphs of each stroke.

The user creates geometry by drawing a

freehand sketch directly on a Tablet-PC screen. The recognition engine cleans up input data and adjusts edges to make sure they meet precisely at common endpoints in order to obtain geometrically consistent figures.

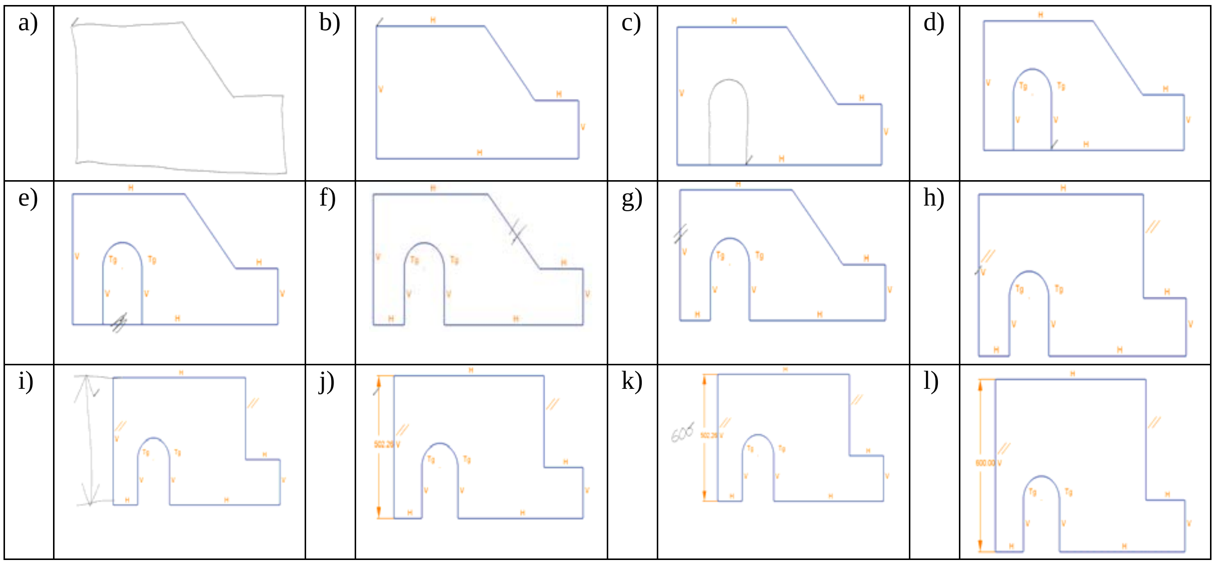

Figure 8. Drawing sequence in ParSketch

Figure 8 shows an example of interaction with *ParSketch*. The user draws the whole contour in 8.a. A single stroke is accepted as input, and is decomposed into six rectilinear and connected strokes. The application shows the beautified version (8.b), and the user adds another complex stroke composed by two segments and one arc (8.c). The geometry is then beautified (8.d).

Figure 8.e shows the use of the scratching gesture to refine the object topology. *ParSketch* interprets this gesture as an order to delete those geometric entities intersecting the smallest quadrilateral which encloses the gesture.

Next, a parallelism constraint is applied by sketching its associated gesture over the two segments to be made parallel (see 8.f, 8.g, 8.h).

Once the desired shape has been obtained, we can proceed with dimensioning. Figure 8.i shows a dimension gesture without any text. *ParSketch* interprets this as a measure command, and shows the current value of that dimension (8.j). The user can change the current dimension value by writing the new value next to the current one. *ParSketch* responds by recalculating its model of the object and displaying the new geometry (8.k and 8.l).

Graphic symbols which represent conventional features are multi-iconic, and are the most challenging problem posed by prescriptive sketches. Strategies such as “multi-agent systems” [14] appear to be the most promising approach for recognising them.

While recognition of handwritten text is now reasonably robust, it benefits from the advantages that text characters are generally grouped together, all of a similar size, and all aligned with one another. *Dimensions* and *hatching* (see Figure 7), amongst others, do not have these advantages. Recognition of such complex standardised symbols is a current challenge.

While single-view approaches are currently the most suitable for automatic 3D reconstruction [6], single-view representations are not the only ones used while designing (ISO 128 and ISO 5456-2 versus ISO 5456 parts 1 and 3). A true CAS system aimed at covering the entire range of conceptual design should be able to deal with both single views (axonometric and the like, usually associated with thinking sketches), and multiple orthographic views, which are more oriented towards prescriptive sketches.

6. Conclusions and Future Work

The engineering community considers sketching to be a vital skill for current and future engineers. However, current commercial CAD applications are not suitable tools for integrating conceptual design (where sketches dominate) and detailed design (where 3-D modelling is central). True CAS tools, oriented towards engineering design, are required.

In previous sections, we have shown that CAS is viable. If is to become an everyday reality, it must in addition be *well-behaved*: it must not present the user with unpleasant surprises. An ideal tool is one which is so predictable that it becomes invisible: use of it is so automatic that the user can concentrate on creating a design without even having to think about how to use the tool.

As well as being well-behaved, a tool must also be useful. Specifically, it must provide all of the functionality which is provided by the WIMP tools which it is to supersede. It must, of course, also provide all of the functionality provided by traditional paper-and-pencil sketching.

We argue that usability is best served by the concept of a *minimal interface*, where we keep user interaction as paper-like as possible. Recognition of standard symbols already in common use also makes an important contribution to usability.

At present, we cannot say with confidence that such tools are more usable than paper and pencil. The problem is one of fine-tuning. In the hands of an experienced sketcher, a pencil is a subtle tool which can be used to produce a wide variety of effects. We are not, as yet, able to reproduce this subtlety while tracking the movements of a computer pen. Further investigation of the physical act of sketching, and how it may be tracked automatically, is required.

In terms of functionality, the capability of Computer Aided Sketching tools to create models of practical interest to engineering designers is limited: curved surfaces are still not supported.

7. Acknowledgements

This work was supported by the Spanish Ministry of Science and Education and the European Union (Project DPI2007-66755) and by the Ramon y Cajal Scholarship Programme.

8. References

[1] R.E. Barr, (2004). *The Current Status of Graphical Communication in Engineering Education.* 34th ASEE/IEEE Frontiers in Education Conference. October 20–23, 2004, Savannah, GA. S1D8–13.

[2] P.J. Booker, (1979), *A History of Engineering Drawing*. Northgate Publishing Co. London.

[3] R.O. Buchal, (2002). *Sketching and Computer-Aided Conceptual Design.* Proceedings of the International Conference on Computer Supported Cooperative Work in Design 7, 112–119.

[4] M. Chamorro-Koc, V. Popovic and M. Emmison, (2008). *Using visual representation of concepts to explore users and designers' concepts of everyday products*. Design Studies 29 (2), 142–159.

[5] P. Company, N. Aleixos, F. Naya, P.A.C. Varley, M. Contero and D.G. Fernandez-Pacheco, (2008). *A New Sketch-Based CAE Pre-Processor*. 6th International Conference on Engineering Computational Technology. Athens, Greece.

[6] P. Company, A. Piquer, M. Contero M. and F. Naya, (2005). *A Survey on Geometrical Reconstruction as a Core Technology to Sketch-Based Modeling*. Computers & Graphics 29(6), 892–904.

[7] M. Contero, F. Naya, J. Jorge and J. Conesa, (2003). *CIGRO: a Minimal Instruction Set Calligraphic Interface for Sketch-Based Modeling*. Lecture Notes in Computer Science. Volume 2669, 549–558.

[8] E.S. Ferguson, (1992). *Engineering and the Mind's Eye*, MIT Press.

[9] J.H. Gong, H. Zhang, B. Jiang and J.G. Sun, (2008). *Identification of sections from engineering drawings based on evidence theory*. Proceedings of the 2008 ACM symposium on Solid and physical modeling, 13–24.

[10] I.J. Grimstead, (1997). *Interactive Sketch Input of Boundary Representation Solid Models*, PhD Thesis, Cardiff University.

[11] Y. Huang, (2008). *Investigating the cognitive behavior of generating idea sketches through neural network systems*. Design Studies 29 (1), 70–92.

[12] B. Jonson, (2002). *Sketching now. International Journal of Art & Design Education*, 21-3, 246–253.

[13] B. Jonson, (2005). *Design ideation: The conceptual sketch in the digital age*. Design Studies 26 (6), 613–624.

[14] R. Juchmes, P. Leclercq and S. Azar, (2005). A freehand-sketch environment for architectural design supported by a multi-agent system. Computers & Graphics 29(6), 905–915.

[15] K. Kanatani, (1990). *Group-Theoretical Methods in Image Understanding*, Number 20 in Springer Series in Information Sciences, Springer-Verlag.

[16] D.C. Ku, S.F. Qin and D.K. Wright, (2006). *A Sketching Interface for 3D Modeling of Polyhedron*. Proc. Eurographics Workshop on Sketch Based Interfaces and Modeling (SBIM06), Sept 03-04, 2006, Vienna, Austria, 83–90.

[17] H. Li, (2006). *nD Polyhedral Scene Reconstruction from Single 2D Line Drawing by Local Propagation*, LNAI 3763 169–197.

[18] D. Lieu and S. Sorby (2008). *Visualization, Modeling And Graphics For Engineering Design*.

[19] S. Lim, S.F. Qin, P. Prieto, D. Wright and J. Shackleton, (2004). *A Study of Sketching Behaviour to Support Free-Form Surface Modelling from On-line Sketching*. Design Studies. 25. 393–413.

[20] H. Lipson and M. Shpitalni, (1996). *Optimization-Based Reconstruction of a 3D Object from a Single Freehand Line Drawing*, Computer Aided Design 28(8), 651–663.

[21] J. Liu, Y.T. Lee, and W.K. Cham, (2001). *Identifying Faces in a 2D Line Drawing Representing a Manifold Object*, IEEE Transactions on Pattern Analysis and Machine Intelligence 24(12), 1579–1593.

[22] V. Livshits and B.Z. Sandler, (1999). *Upstairs/Downstairs in Technical Education: The Unsettling Effects of Computerization*. International Journal of Technology and Design Education 9, 73–84.

[23] R.R. Martin, P.A.C. Varley and H. Suzuki, (2005). *Perpendicularity as a Key to Interpreting Line Drawings of Engineering Objects*, Proc. Digital Engineering Workshop: 5th Japan-Korea CAD/CAM Workshop, 115–120.

[24] M. Masry and H. Lipson, (2005). *A Sketch-Based Interface for Iterative Design and Analysis of 3D Objects*. in ed. J. F. Hughes and J. A. Jorge, Sketch-Based Interfaces and Modelling, Eurographics Symposium Proceedings,109–118.

[25] A. Menezes and B. Lawson, (2006). *How Designers Perceive Sketches*. Design Studies 27 (5), 571–585

[26] F. Naya., M. Contero, N. Aleixos and P. Company (2007). ParSketch: A Sketch-Based Interface for 2D Parametric Geometry Editor. Lecture Notes in Computer Science. HCII 2007. Vol. 4551, 115–124.

[27] L. Olsen, F. Samavati and J.A. Jorge, (2008). *A Taxonomy of Modeling Techniques using Sketch-Based Interfaces*, Eurographics 2008 State of the Art Reports, Crete.

[28] A. Piquer, R.R. Martin and P. Company, (2004). *Skewed Mirror Symmetry for Depth Estimation in 3D Line-Drawings*. Lecture Notes in Computer Science. GREC 2003 Post-proceedings. Volume 3088, 138–149.

[29] A.T. Purcell and J.S. Gero, (1998). *Drawings and the design process*. Design Studies 19 (4), 389–430.

[30] L.G. Roberts, (1963). *Machine Perception of Three-Dimensional Solids*. PhD Thesis, MIT.

[31] A.T. Rose, (2005). *Graphical Communication Using Hand-Drawn Sketches in Civil Engineering*. Journal of Professional Issues in Engineering Education and Practice. Volume 131, Issue 4, 238–247.

[32] Y. Suh, (2007). *Reconstructing 3D Feature-Based CAD Models by Recognizing Extrusions from a Single-View Drawing*, Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference.

[33] S.J. Tate and G.E.M. Jared, (2003). *Recognising symmetry in solid models*. Computer-aided Design 35 (7), 673–692

[34] P.A.C. Varley, (2003). *Automatic Creation of Boundary-Representation Models from Single Line Drawings*, PhD Thesis, University of Wales.

[35] P.A.C. Varley, (2005). *Problems For Line Labelling: A Test Set of Drawings of Objects with Higher-Valency Vertices*, International Journal of CAD/CAM 5 (1).

[36] P.A.C. Varley and P.P. Company, (2008). *A New Algorithm for Finding Faces in Wireframes*, in preparation

[37] P:A.C. Varley, R.R. Martin and H. Suzuki, (2005). *Frontal Geometry from Sketches of Engineering Objects: Is Line Labelling Necessary?*, Computer Aided Design 37 (12), 1285–1307.

[38] R. Vidal, E. Mulet and E. Gomez-Senent, (2004). *Effectiveness of the means of expression in creative problem-solving in design groups*. Journal of Engineering Design, Volume 15, Number 3, 285–298.

[39] W. Wang and G.G. Grinstein, (1993). *A Survey of 3D Solid Reconstruction from 2D Projection Line Drawings*, Computer Graphics Forum 12(2), 137–158.